# Summary of New Features in Magma V2.13

**July 2006**

## 1 Introduction

This document provides a terse summary of the new features installed in Magma for release version V2.13 (July 2006).

Previous releases of Magma were: V2.12 (June 2005), V2.11 (May 2004), V2.10 (April 2003), V2.9 (May 2002), V2.8 (July 2001), V2.7 (June 2000), V2.6 (November 1999), V2.5 (July 1999), V2.4 (December 1998), V2.3 (January 1998), V2.2 (April 1997), V2.1 (October 1996), V2.01 (June 1996) and V1.3 (March 1996).

## 2 Summary

### Groups

- *Finitely-Presented Groups:* An algorithm due to Derek Holt for testing whether two finitely presented groups are isomorphic has been implemented in Magma by Derek. The algorithm uses the Knuth-Bendix procedure to enumerate elements.
- *Finitely-Presented Groups:* Machinery for classifying metacyclic $p$-groups developed by Eamonn O'Brien and Michael Vaughan-Lee has been included.
- *Matrix Groups over Finite Fields:* Constructive recognition of a matrix group as $SL(3, q)$ has been provided by Eamonn O'Brien. The corresponding code for $SL(2, q)$ has been improved.
- *Matrix Groups over Finite Fields:* It is now possible to recognise the Suzuki and Ree groups in various matrix representations using a package developed by Hendrik Bäärnhielm. This makes use of the code for recognising $SL(2, q)$. The package also contains functions to compute Sylow $p$-subgroups for these two families of groups.
- *Matrix Groups over Finite Fields:* It is now possible to construct a Sylow $p$-subgroup of any classical group using a package developed by Mark Stather. Functions are also included for computing normalisers and solving the conjugacy problem for Sylow subgroups.

### Basic Rings

- *Real and Complex Numbers*: Support for real numbers has been improved in this release by integrating in the latest version of the MPFR library. This includes new implementations of many functions (such as the gamma function) which are faster and more stable than the previous implementations.

### Linear Algebra and Module Theory

- *Lattice Reduction:* A new implementation of LLL reduction of integer lattices has been undertaken by Damien Stehlè and is based on the Nguyen–Stehlé floating-point algorithm. The LLL and LLLGram algorithms are now guaranteed both to complete and to produce LLL-reduced bases. The new algorithm is more efficient than the previous one, sometimes dramatically so.

- Very great speedups have been achieved for the fundamental matrix algorithms over small and moderately-sized finite fields, and new fast modular algorithms for Hermite and Smith normal form computation have been introduced.

### Extensions of Rings

- *Series Rings:* A uniform interface for series rings and local rings has been defined. Factorization of polynomials over local rings has been generalised for polynomials over series rings. Extensions of series rings can be constructed.

- *Number Fields:* A new algorithm for the computation of Galois groups of extensions of the rationals and of square-free polynomials over the integers has been implemented. This is the first ever degree-independent implementation. It has already been used in degrees up to 64.

  Support for infinite places of relative extensions has been added as well.

- *Algebraic Function Fields:* Functions have been added to perform Weil Descent on Artin-Schreier extensions of the rational function field in characteristic $p$. This generalises the Gaudry-Heß-Smart explicit descent method in characteristic 2. The code was contributed by Florian Heß.

- *Algebraically Closed Fields:* Algebraically closed fields may now be defined over finite fields and rational function fields over finite fields or the rational field, as well as the rational field.

### Algebras

- *Orders of Associative Algebras:* These orders have been reimplemented and expanded. Functionality for orders defined over orders of number fields has been provided by representing basis information as a combination of a matrix and coefficient ideals. Ideals of these orders can now be constructed.

- *Orders of Quaternion Algebras:* These orders now inherit from the orders of associative algebras instead of the associative algebras themselves.

- *Quaternion Algebras:* It is now possible to work with quaternion algebras over number fields. It can be determined whether an associative algebra is quaternion. Matrix rings can be constructed from quaternion algebras.

**Lie Theory**

- *Coxeter groups:* Weight orbits and the dominant weight in an orbit can now be computed – these tools are useful for understanding representations. Several other functions have been speeded up.

- *Root data and root systems:* Non-reduced root data and systems, such as type $BC_n$, can now be constructed. Extended root data are also available – these contain the combinatorial data necessary to construct twisted groups of Lie type. A new category, `RootDtmSprs`, provides a sparse representation for classical root data. This requires much less memory and makes it possible to work with root data of very large rank. Morphisms between root data, including isomorphisms, fractional morphisms and dual morphisms have been implemented.

- *Groups of Lie type:* Twisted groups of Lie type are now available (these include unitary groups). A large speed up in element operations has been achieved by implementing a new algorithm called collection from outside. A new method for element multiplication in classical groups of Lie type, based on sparse root data, has been implemented, resulting in big memory savings.

**Algebraic Geometry**

- *General Schemes:* Point searching using the Elkies ANTS-IV $p$-adic method is now implemented for general schemes defined over the rationals.

- *Algebraic Curves:* Functionality has been added for ordinary plane curves and to produce random curves with genus $\leq 13$. The ordinary curve functionality includes much faster computation of canonical maps/images and much faster parametrization of rational curves.

- *Surfaces:* Functions for the parametrization of degree 6, 8 and 9 Del Pezzo surfaces over $\mathbf{Q}$ have been added. The degree 8 and 9 packages were contributed by Jana Pilnikova.

**Arithmetic Geometry**

- *Elliptic Curves over the Rationals:* A new package has been included for performing a full 3-descent on any elliptic curve over $\mathbf{Q}$. This involves computing the 3-Selmer group, and then representing the elements as plane cubic curves.

- *Elliptic Curves over Function Fields:* A new package has been included for elliptic curves defined over algebraic function fields (function fields of curves). The package includes an implementation of Tate's algorithm, and height machinery, in this generality. It also includes, in lesser generality, routines for computing the $L$-function, the 2-Selmer group, and the Mordell-Weil group.

- *Elliptic Curves over finite fields:* Functions to perform Gaudry-Heß-Smart Weil descent on elliptic curves in characteristic 2 have been added. The core of the code was contributed by Florian Heß.

- *Genus One Models:* A package has been contributed by Tom Fisher dealing with invariant theory of genus one normal curves of degrees $2, 3, 4$ and 5, and arithmetic applications of this.

- *Hyperelliptic Curves over Finite Fields:* A package has been contributed by Hendrik Hubrechts for counting points/ computing zeta functions of Jacobians of hyperelliptic curves lying in parametrized families, following the deformation method of Lauder.

- *Modular Abelian Varieties:* A package has been contributed by Jordi Quer for determining the endomorphism algebra, and the fields of definition, of a building block of a modular abelian variety.

**Coding Theory**

- *LDPC Codes:* A module for constructing, decoding and analyzing Low Density Parity Check codes has been developed. As well as iterative decoding, the module includes simulation tools such as density evolution.

- *Algebraic-geometric Codes:* Machinery for decoding algebraic-geometric codes up to the Goppa designated distance has been added.

- *McEliece Cryptosystem:* The best published decoding attacks on the McEliece cryptosystem together with improved attacks have been implemented. These include the attacks developed by McEliece, Lee & Brickell, Leon, Stern and Canteaut & Chabaud as well as generalized combinations of attacks.

# 3 Removals and Changes

This section lists the most important changes in Version 2.13. Other minor changes are listed in the relevant sections.

– The `LLL` and `LLLGram` functions have been changed so that they now produce a guaranteed LLL-reduced basis by default. (In particular, the output is no longer sorted.) Consequently the results may be slower on some problems; the old behaviour may still be obtained via the intrinsics `BasisReduction` and `GramReduction`. Additionally, the possible parameters have also changed.

# 4 Documentation

New chapters in the Handbook for V2.13 (with their chapter numbers) are:

- Matrix Groups Over General Rings (replaces Matrix Groups) (19)
- Matrix Groups Over Finite Fields (20)
- Orders of Associative Algebras (72)
- Elliptic Cuves over Finite Fields (102)
- Elliptic Cuves over Function Fields (103)
- Models of Curves of Genus 1 (104)
- Algebraic-geometric Codes (124)
- Low Density Parity Check Codes (125)

# 5 Language and System Features

New Features:

- Magma now offers facilities to read and write binary data. In previous versions, reading and writing was restricted to string types, which have trouble handling non-printable characters. A new type has been introduced, with semantics similar to a sequence of integers, but the implementation of which is tuned for character arrays.
- Functions, procedures, and intrinsics, can now be *variadic*; that is, they can take in a variable number of arguments.
- The maximum number of return values for an intrinsic package function has been increased from 5 to 256.
- Exception handling has been implemented, using the familiar notion of try/catch statements found in many other languages, such as Java, C++, and Python. Currently, Magma supports the catching of two kinds of errors: system errors and user errors. The user has the ability to attach arbitrary data to error objects through the use of attributes.
- Magma now has an "eval" keyword, similar to that found in languages like Python and Perl. This allows the evaluation of a string as a piece of Magma code. For instance, if `s` is a string with value "1+1", then `x := eval s` assigns to `x` the value 1. This feature allows a great deal of runtime flexibility, and is useful, for instance, in the implementation of databases of objects.

# 6 Aggregates

Bug Fixes:

- Removal of multiset elements no longer causes the multiplicities of other elements to change.

# 7 Groups

## 7.1 Matrix Groups Over General Rings

New Features:

- `IsAbelian`, `IsElementaryAbelian`, and `IsCyclic` now work for infinite matrix groups.

Bug Fixes:

- A bug was fixed in the test for finiteness of a group defined over a large degree extension of $\mathbf{Q}$.

## 7.2 Matrix Groups over Finite Fields

Changes:

- Constructive recognition of a matrix group as $SL(3, q)$ has been provided by Eamonn O'Brien. The corresponding code for $SL(2, q)$ has been improved.
- The original code of Alice Niemeyer for finding a form fixed by a classical group has been replaced by code written by Derek Holt. This code overcomes a number of errors and omissions in the Niemeyer version.
- A revised version of code for recognising classical groups has been provided by Alice Niemeyer. It fixes known problems and uses the Holt code for forms.
- A more efficient version of the Monte-Carlo program for recognising quasi-simple groups developed by G Malle and E O'Brien has been installed.

New Features:

- Code developed by Henrik Baarnhielm for recognising the Suzuki and Ree groups in various matrix representations is now included. The package also has functions to compute the Sylow subgroups for these two families of groups.
- A package developed by Mark Stather for computing the Sylow subgroups of the classical groups (given as matrix groups) has been included. This can also compute normalisers and solve the conjugacy problem for Sylow subgroups.

## 7.3 Finite Soluble Groups

Changes:

- The use of the obsolete Stackhandler Memory Manager has been completely removed from the soluble groups module.

New Features:

- The Leedham-Green algorithm for computing the center of a soluble group defined by a pc-presentation has been implemented.
- A more efficient algorithm for the calculation of centralisers has been installed.

## 7.4 Finitely Presented Groups

New Features:

- An algorithm due to Derek Holt for testing whether two finitely presented groups are isomorphic has been implemented in Magma by Derek. The algorithm uses the Knuth-Bendix procedure to enumerate elements.

# 8 Basic Rings

## 8.1 Integer Ring

New Features:

- The GMP 4.2.1 multiprecision integer library is now linked in.
- New function `Normalize` for residue ring elements.

## 8.2 Real and Complex Fields

New Features:

- Support for real numbers has been improved in this release by integrating in the latest version of the MPFR library. This includes new implementations of many functions (such as the gamma function) which are faster and more stable than the previous implementations.

# 9 Linear Algebra and Module Theory

## 9.1 Matrices

New Features:

- Very great speedups have been achieved for the fundamental matrix algorithms over small and moderately-sized finite fields. In particular, matrix multiplication over GF(q) for $q = 3, 4, 5, 7, 8, 16$ uses a new fast packed representation.
- New fast modular algorithms for the computation of Hermite or Smith normal forms has been developed. See `http://tinyurl.com/z68bu` for a webpage which gives timings involving the new Hermite algorithm.
- New function `RandomMatrix` for convenient construction of random matrices over finite rings.

## 9.2 Modules over Dedekind domains

The pseudo matrix structure underlying the modules have been made available.

New Features:

- A pseudo matrix can be constructed from a sequence of ideals and a matrix. These coefficient ideals and the matrix can be returned from the pseudo matrix as well as the order the pseudo matrix is over and the length and dimension of the pseudo matrix. Pseudo matrices can be compared for equality.
- Pseudo matrices can be transposed. A `HermiteForm` of a pseudo matrix can be computed and 2 pseudo matrices can be vertically joined.

# 10 Commutative Algebra

New Features:

- For homogeneous ideals, the Radical and Equidimensional Decomposition computations have been reimplemented using a more homological approach. These are now generally faster — much more so in some cases.
- New function `MonomialBasis` for quotient rings (affine algebras).

# 11 Extensions of Rings

## 11.1 Algebraic Number Fields

New Features:

- Support for infinite places of (relative) extensions has been added.
- Computation of Galois groups for absolute extensions and square-free integer polynomials is now possible without any degree limitations. This replaces the old method by Geißler.
- Functionality to compute arbitrary fields in the normal closure of number fields has been rewritten to complement the new Galois group computations.
- A places/divisor based interface to ray class groups has been added. In particular, defining modules including infinite places becomes much cleaner.

Bug Fixes:

- The fields returned by `SubfieldLattice` are now identical to those reported by `Subfields`. In the case of non-monic defining polynomials this was not the case previously.

## 11.2 Algebraically Closed Fields

New Features:

- Algebraically closed fields may now be defined over finite fields and rational function fields over finite fields or the rational field, as well as the rational field.

## 11.3 Quadratic Fields

Bug Fixes:

– The map between quadratic forms and ideals in non-maximal orders has been fixed.

## 11.4 Abelian Extensions

New Features:

– A new interface that allows places and divisors to be used to define class fields and ray class groups has been added.

– It is now possible to use `subset`, `meet` and `'*'` on abelian extensions of the same base field.

## 11.5 Algebraic Function Fields

Removals and Changes:

– The old intrinsic `IsIsomorphic` which finds isomorphisms of function fields $E$ and $F$ over a common $\mathbf{Q}(t)$ base field has been renamed `IsIsomorphicOverQt`. This prevents the clash with the version of `IsIsomorphic` which finds isomorphisms between more general fields $E$ and $F$ extending a given isomorphism of their constant fields.

New Features:

– A function `WeilDescent` which performs Weil Descent on Artin-Schreier extensions of the rational function field over finite fields has been included. This generalises the explicit descent method of Gaudry, Heß and Smart in characteristic 2. If the constant field of the function field $E$ is $K$ and $k$ is the subfield of $K$ to be descended to, the result is a function field $F$ with constant field $k$ together with a divisor map from places/divisors of $E$ to divisors of $F$.

– There are related helper functions: `ArtinSchreierExtension` to generate the field $E$ from parameters; `WeilDescentGenus` and `WeilDescentDegree` to compute the genus and degree (over its base field) of $F$ before performing the descent.

– It is now possible to form completions in global function fields at places of degree greater than 1.

Bug Fixes:

– The calculation of 2 generators of an ideal has been improved by the increased use of the algorithm of Belabas especially when random elements of the bottom coefficient ring are not available.

## 11.6 Newton Polygons

Bug Fixes:

– A bug in `IsInterior` when the polygon was a line and defined by all its faces has been fixed.

## 11.7 Series Rings

New Features::

- A few intrinsics have been added in order to make the series rings compatible with the $p$-adic rings and fields. These are `ResidueClassField`, `ChangePrecision` and `UniformizingElement`. It is now also possible to use `HenselLift` with polynomials over series rings.
- Now that series rings share a uniform interface with $p$-adic rings the `Factorization` algorithm implemented for $p$-adic rings and their extensions can be used to factor polynomials over series rings over finite fields also.
- Series rings over finite fields can now also be extended. Extensions must be either unramified or totally ramified and can be made using `UnramifiedExtension` and `TotallyRamifiedExtension`.
- Extensions of series rings over finite fields have type `RngSerExt`. They support the uniform interface of $p$-adic rings and series rings. This interface includes `Precision`, `UniformizingElement`, `ChangePrecision`, `ResidueClassField`, equality of rings, and `Valuation`, arithmetic and predicates on elements. For extensions it also includes `InertiaDegree` and `RamificationIndex`. Additionally `CoefficientRing` and `DefiningPolynomial` are supported.
- Polynomials over extensions of series rings can be factored using the `Factorization` algorithm available for $p$-adic and series rings.
- Unramified extensions of series rings can be converted using `OptimizedRepresentation` to an isomorphic series ring.

# 12 Differential Rings

## 12.1 Differential Rings

Removals and Changes:

- Infinite precision is no longer denoted using -1, `Infinity()` is used instead. This also effects intrinsics taking `Precision` parameters.

# 13 Lattices and Quadratic Forms

## 13.1 Lattice Reduction

The `LLL` and `LLLGram` routines have been almost completely rewritten.

New Features:

- Correctness. The default output of these routines is always LLL-reduced for the input pair of factors. To achieve this, the `FinalSort` option has been turned off by default.
- Termination. The calls to the `LLL` and `LLLGram` routines should always terminate.
- A new LLL factor, `Eta`, has been introduced. It is used for the size-reduction property: the Gram-Schmidt coefficients $\mu_{i,j}$ of the output basis satisfy $|\mu_{i,j}| \leq \texttt{Eta}, \forall i > j$. By default, `Eta` is set to 0.501.

– To avoid any ambiguity, the `Sort` option has been renamed `InitialSort`.

– An `EarlyReduction` option has been added: when a new vector is visited, the other vectors are sometimes size-reduced as with respect to the already reduced vectors.

– The traditional Lovász swapping condition may now be replaced by the so-called Siegel swapping condition, with the `SwapCondition` option.

– A `Fast` option has been added: the system will try to choose the best user parameters in order to terminate as fast as possible.

– The `LLLGram` routine can take as input any symmetric matrix, since it makes use of Simon's variant of the Lovász swapping condition.

– The following parameters are now deprecated: `InitialDelta`, `DeltaSteps`, `FPBlock`, `Large` and `UnderflowCheck`.

– The zero lattice is now supported.

# 14 Algebras

## 14.1 Quaternion Algebras

Orders of quaternion algebras now inherit from orders of general associative algebras (`AlgAssVOrd`) instead of from the structure constant associative algebras. Ideals of orders of quaternion algebras inherit from the ideals of orders of general associative algebras (`AlgAssVOrdIdl`).

Orders of quaternion algebras can now be defined over orders of number fields. These orders and ideals however use the general order and ideal types `AlgAssVOrd` rather than the specific quaternion order and ideal types.

Removals and Changes:

– Orders and ideals of orders of quaternion algebras no longer inherit from associative algebras so will no longer have all the functionality of associative algebras. They will instead have all the functionality of `AlgAssVOrd` and `AlgAssVOrdIdl`.

– Ideals of orders of quaternion algebras are no longer structures but are elements of a power ideal. They now have their own type `AlgQuatOrdIdl`. Bases of ideals are now returned as sequences of elements of an algebra rather than of ideals (ideals no longer have elements). Some output may appear different because it is with respect to a different basis. `Basis` and `BasisMatrix` can accept a second argument of a structure the output will be with respect to.

– Functionality for the old `AlgQuatOrd` has been split between `AlgQuatOrd` and `AlgQuatOrdIdl` as appropriate.

– The `Composite` intrinsic for ideals has been removed.

– `IsRamified` for integer primes in quaternion algebras over the rational field has been replaced. The order of the arguments has been swapped, the prime now being the first argument.

New Features:

– A general associative algebra can be tested if it is a quaternion algebra by `IsQuaternionAlgebra`. A standard representation is returned.

– Given a zero divisor in a quaternion algebra, one can compute an isomorphism to the matrix ring by the command `MatrixRing`.

– The `IsDefinite` intrinsic now works for quaternion algebras over any number field and for orders over number rings.

– Full functionality for the Hilbert symbol has been added. One can compute the set of ramified places of a quaternion algebra.

– A quaternion algebra can be constructed by specifying any even set of noncomplex places of a number field.

– Embeddings for quadratic fields into quaternion algebras and quadratic orders for quaternion orders has been added in the intrinsics `IsSplittingField`, `HasEmbedding`, and `Embed`.

– The new intrinsic `pMatrixRing` computes a local splitting of a quaternion algebra over a number field at an unramified prime.

– A tame order and a maximal order of a quaternion algebra over a number field (or containing a given order) can be computed, as well as $p$-maximal orders. Orders can be tested for maximality and $p$-maximality.

– For definite quaternion algebras (over totally real number fields), the intrinsic `Enumerate` now lists elements in an ideal or order with bounded norm (or bounded in a box with respect to a Minkowski embedding). See also `LatticeVectorsInBox` which works for a general lattice.

– One can compute an `OptimizedRepresentation` of either a quaternion algebra or order; the result will tend to have much "smaller coefficients".

– For orders of quaternion algebras over totally real number fields, one can compute a reduced basis via `ReducedBasis`. This basis is either the canonical one if the algebra is totally definite, and is a Minkowski- like embedding otherwise.

– For a definite quaternion order (now also over an order of a number field), the intrinsic `UnitGroup` returns the group of units of an order modulo the group of units of the base ring; returns an abstract group as well as a map to the order.

– One can test if two (left or right) ideals of a quaternion algebra (now also of a number field) are isomorphic, as well as a complete set of (left or right) ideal classes. In particular, one can test if an ideal is principal and, if so, compute a generator.

Bug Fixes:

– Zero input on the right hand side of the `QuaternionAlgebra` constructor is checked for and an error occurs.

## 14.2 Orders of Associative Algebras

Orders of general associative algebras have been rewritten and extended. Some specific functionality is provided for orders defined over orders of number fields.

New Features:

– Orders of associative algebras can now be created over orders of number fields by specifying a pseudo-basis. This pseudo-basis consists of a matrix and a sequence of coefficient ideals.

– Orders can be constructed by specifying any generating set of algebra elements. One can adjoin an algebra element to an order or compute the sum of two orders.

- Maximal orders can be computed as well as $p$-maximal orders over number rings.
- Information regarding the basis of the order can be returned as a `PseudoMatrix` or `PseudoBasis`.
- Orders can now be compared to determine whether they are equal. It can also be determined whether an algebra element is in the order.
- Basic functionality for orders, including computing degree, trace-zero subspace, and discriminant.
- Orders now have their own elements of type `AlgAssVOrdElt`. Elements can be added, subtracted, negated, multiplied, multiplied by scalars, divided and powered as well as tested for equality and for equality with zero. The intrinsics `LeftRepresentationMatrix`, `RightRepresentationMatrix`, `MinimalPolynomial`, `Norm`, `Trace`, `Conjugate` and `Eltseq` are also supported.
- Ideals of orders of associative algebras can be formed. These have type `AlgAssVOrdIdl`. Ideals can be left, right or two-sided. A basis of an ideal can be retrieved and left and right orders (multiplicator rings) can be computed, as well as the colon of two ideals. Addition and multiplication of compatible ideals is also available.
- The `Algebra` an ideal is contained in is available as well as the `Order` the ideal was created as an ideal of.
- `Basis` and `BasisMatrix` of an ideal can be retrieved with respect to a given order or algebra. Basis information can also be accessed as a `PseudoMatrix` or `PseudoBasis`.
- Ideals can be tested for being left or right ideals. They can be created by multiplying an order by an element and compared to determine equality. One can test for containment of an element or inclusion of ideals as well as compute the (reduced) norm of an ideal.

# 15 Lie Theory

## 15.1 Root Systems and Root Data

Removals and Changes:

- The extraspecial signs are now assigned to a root datum upon creation and cannot be changed afterwards.
- Some changes has been made to intrinsics which return roots as vectors. Previously, the returned vectors were over integers in some cases, over rationals in other cases. Now the returned roots are always vectors over the field of rational numbers.
- Optional parameter `Basis` has been added to all intrinsics returning roots as vectors or taking roots as vectors as arguments, to indicate the basis, with respect to which the vectors are built.
- Some changes have been made to internal handling of (co)root lattices and (co)root spaces. The full root lattice $X$ associated with a root datum is now returned by `FullRootLattice`, the sublattice spanned by simple roots is returned by `RootLattice`, and the root space $X \otimes \mathbf{Q}$ is returned by `RootSpace`. The coroot lattices and coroot space can be obtained in a similar way.
- Creation of root subdata and root subsystems has been improved and the constructor `sub<..>` can now be used to create them.

New Features:

– Non-reduced root data and systems of type $BC_n$ are now supported. Reducednes of a root datum or system can now be checked by using the intrinsic `IsReduced`.

– The subsystem or subdatum consisting of indivisible roots of any root system or datum can be constructed by `IndivisibleSubsystem` or `IndivisibleSubdatum`, respectively.

– Indivisibility of roots can be checked by using `IsIndivisibleRoot`.

– A new category for sparse root data `RootDtmSprs` has been implemented. This requires much less memory, as neither roots nor constants associated with a root datum are stored in memory, but always computed when required. Due to repeated computation of constants and roots, this is slower than the standard, dense, representation of root data. But the sparse representation is unavoidable for large ranks. For example, the root datum of type $B_{1500}$ requires about 122MB of memory using the sparse representation and can't be created on a machine with 2GB of memory using the dense representation.

– Extended (twisted) root data can now be constructed. This corresponds to Tits indices and carries information associated with the root datum of a twisted group of Lie type.

– Relative root datum can be computed from an extended root datum.

– Morphisms between root data can be constructed. This includes dual morphisms, isomorphisms and fractional morphisms.

## 15.2 Coxeter Groups as Permutation Groups

New Features:

– It is now possible to compute dominant weights and weight orbits.

## 15.3 Groups of Lie Type

Changes:

– Element operations in groups of Lie type have been dramatically improved. Collection algorithms have been implemented in the C kernel replacing the previous package code, resulting in large speed-ups.

– New algorithms for element operations have been implemented. The available algorithms in the current release include:

  – Collection To Left.
  – Collection From Left.
  – Collection From Outside (new).
  – Symbolic Collection From Left.
  – Symbolic Collection From Outside (new).
  – Symbolic Collection using direct formulas for classical types (new).

– The algorithm used for element operations can be specified by the user. By default, the fastest algorithm for the given group is chosen.

– Properties and constants of groups of Lie type, that do not depend on the base ring of the group, are now stored in the associated root datum. This speeds up subsequent creation of groups of Lie type having the same root datum over different base rings.

New Features:

– Twisted groups of Lie type can now be constructed.

# 16 Algebraic Geometry

## 16.1 Schemes

Removals and Changes:

– The `Check` parameter on the creation of maps between schemes now controls whether the defining polynomials define a map into the codomain of the scheme. The `CheckInverse` parameter now controls whether the inverse is checked to be an inverse. Both are `true` by default. If `Check` is set to `false` then the checking of the inverse will only be done if `CheckInverse` is set to `true`.

– Some operations involving function fields of schemes may be faster due to function fields knowing they are fields and not having to determine this.

New Features:

– A function `HeightOnAmbient` has been added, for calculating the height of a point on a general variety in affine or projective space. The point may be defined over the rationals, a number field, or a function field.

– A nontrivial algorithm to search for points on general schemes over the rationals has been implemented under the name `PointSearch`. The scheme can be in any affine or non-weighted projective space. This uses a $p$-adic algorithm: first find points locally modulo a small prime (or two small primes), then lift these $p$-adically, and then see if these give global solutions. Lattice reduction is used at this stage, and this makes the method far more efficient than a naive search. In fact, it becomes more efficient as the dimension of the ambient space increases; for instance, the asymptotic time to find points up to absolute height $H$ on a curve in $\mathbf{P}^d$ is $O(H^{2/d})$.

– Functions to determine the existence of and explicitly construct parametrisations over $\mathbf{Q}$ of Del Pezzo surfaces of degrees 6, 8 and 9. The main functions are `ParametrizeDegree9DelPezzo`, `ParametrizeDegree8DelPezzo` and `ParamatrizeDegree6DelPezzo`. The algorithms used are based on working with the Lie algebra of the automorphism group of the surface to reduce the explicit construction of a parametrization to at most solving a norm equation over $\mathbf{Q}$. The third of these functions has an option to determine only the existence of a parametrization using local solubility which is much faster.

– Functions for generating Degree 6 Del Pezzo surfaces having a given 2-dimensional torus as automorphism group. These are `Degree6DelPezzoTypeX` where X is `2_1`, `2_2`, `2_3`, `3`, `4` or `6` depending on the torus type.

– Function `RationalPointsByFibration` for finding all points of a general scheme $X$ over a finite field. This is much more efficient than the old `RationalPoints` which now calls the new function by default in most cases. The method is to sum up the points in the zero-dimensional fibres of a finite map of $X$ to a hyperplane or, more generally, a well chosen hypersurface.

Bug Fixes:

– Coercion into function fields of schemes has been improved (although possibly restricted) by delaying the attempt to coerce into the base ring of the scheme.

## 16.2 Algebraic Curves

Removals and Changes:

- The functions `CanonicalLinearSystem` and `AdjointLinearSystem` which return the linear system of polynomials giving the canonical map or the more general degree $d$ adjoint linear system for a projective plane curve $C$, now use a faster, more efficient computational method when $C$ is determined to be an ordinary curve (see below). This relies on a direct computation of the full adjoint ideal.

New Features:

- There are new functions for computing the automorphism group of a general curve and isomorphisms between curves. This transfers over the existing functionality for algebraic function fields.
- `Automorphisms` returns the sequence of all, or up to a given number of, automorphisms of a curve.
- `AutomorphismGroup` returns the full automorphism group of a curve as a generic group along with a map from the group to the actual isomorphisms given as scheme maps.
- `IsIsomorphic` determines whether two curves are isomorphic and finds an explicit isomorphism between them. `Isomorphisms` returns a sequence of all, or up to a given number of, isomorphisms between two curves.
- Functions have been added to generate random curves of given genus $\leq 13$ or with specified singularities over finite fields and $\mathbf{Q}$.
- The main function is `RandomCurveByGenus` which takes a field $K$ and genus $g$ as arguments. If $g \leq 10$, it returns a random plane curve with only nodes as singularities. For $g > 10$, it returns a curve in $\mathbf{P}^3$.
- `RandomNodalCurve` returns a random plane curve with only nodes as singularities. The degree and number of nodes is specified by the caller.
- `RandomOrdinaryPlaneCurve` returns a random plane curve with only ordinary singularities. The degree of the curve, and the number of singularities of multiplicity 2,3,4,... are specified by the caller.
- There are a number of new functions that deal with ordinary plane curves (ones with only ordinary singularities). These rely on the computation of the *adjoint ideal* for ordinary curves more quickly and efficiently than by generic function field methods or by the general resolution of singularities for plane curves. The $d$-th graded part of this ideal gives the degree $d$ adjoint linear system of the curve. For a plane curve of degree $d$, the degree $d-3$ adjoint linear system is the canonical linear system of polynomials that give the canonical map.
- There are functions to determine whether a plane curve is ordinary or nodal, compute the adjoint ideal of such a curve and to return the adjoint linear system of a specified degree given the adjoint ideal.
- A `CanonicalImage` function has been added. This gives the canonical embedding of a plane curve and is much faster than using the general image machinery for the canonical map.

# 17 Arithmetic Geometry

## 17.1 Rational Curves and Conics

Removals and Changes:

– The `Parametrization` functions for a genus 0 curve $C$ use an improved method when $C$ is determined to be an ordinary plane curve. This utilises the functionality for ordinary curves described earlier and the new parametrization for rational normal curves described below.

New Features:

– A routine for solving diagonal conics over number fields, using a version of Legendre's method is provided under the name `LegendresMethod`. This is an alternative to the standard approach of solving such problems using `NormEquation`, and the solutions obtained are often far simpler. This routine will be improved in future releases.

– `ParametrizeRationalNormalCurve` is a new function that gives a fast method of parametrizing a rational normal curve $C$ in projective space over any field. If $C$ lies in $\mathbf{P}^d$ for $d$ odd then the function returns a scheme isomorphism from the projective line $\mathbf{P}^1$ to $C$. If $d$ is even the isomorphism is from a plane conic to $C$. The function uses the geometric method of adjoint maps rather than the function field machinery.

## 17.2 Elliptic Curves

New Features:

– The database of elliptic curves of small conductor constructed by John Cremona has been updated to include all curves having conductor up to 130 000.

### 17.2.1 Mordell–Weil groups

New Features:

– A routine for computing the `Saturation` (at primes up to some bound) of the group generated by a given list of points has been added.

### 17.2.2 Heegner Points

A considerable amount of information about Heegner points can now be obtained. Previously in Magma, Heegner points were seen primarily as a tool for computing points on elliptic curves.

New Features:

– For a given elliptic curve $E$ and discriminant $D$ `Heegner Forms(E,D)` computes a set of points in the upper half plane which represent a Galois orbit of CM points on $X_0(N)$ where $N$ is the conductor of $E$. The images of these points on $E$ under the modular parametrisation are computed (over their field of definition, which is a subfield of the class field of $\mathbf{Q}(\sqrt{D})$) by `CMPoints(E,D)`. The discriminant is not required to be a fundamental discriminant. (For a fundamental $D$, the sum of these points on the elliptic curve is a standard Heegner point in $E(\mathbf{Q})$).

### 17.2.3 Three-Descent

This is a new package for performing a full 3-descent for any elliptic curve over $\mathbf{Q}$. There are two stages to the descent process: first computing the 3-Selmer group, and then representing its elements as plane cubic curves with maps to the given elliptic curve. There is functionality for producing nice models of these curves. The package also contains a completely separate implementation of "descent by 3-isogeny". Large parts of the code were written by Tom Fisher and Michael Stoll. Features:

- The `ThreeSelmerGroup` of any elliptic curve over $\mathbf{Q}$ can be computed. An abstract group, together with a map to the relevant affine algebra, is returned. This is an implementation of the algorithm given by Schaefer and Stoll in *How to do a p-descent on an elliptic curve*, Transactions of the AMS, **356** No. 3, 2004.
- Any element in the `ThreeSelmerGroup` of an elliptic curve $E$, or more generally any suitable element of the relevant affine algebra, can be represented as a plane cubic curve with covering map to $E$ (defined over $\mathbf{Q}$). The intrinsic that does this for a given Selmer element is `ThreeDescentCubic`. Alternatively, the whole process may be performed together by calling `ThreeDescent(E)`. The algorithm for this is joint work by Cremona, Fisher, O'Neil, Simon and Stoll (to appear).
- The reverse process, of starting with a plane cubic curve $C$ and obtaining its Jacobian $E$ and the element that $C$ represents in the 3-Selmer group of $E$, can also be carried out. The intrinsics are `Jacobian` and `ThreeSelmerElement`. Also, given two plane cubics with the same Jacobian, one can form their sum in the Weil-Chatelet group using `AddCubics` (the computation goes via their `ThreeSelmerElement`'s).
- There are intrinsics `ThreeTorsionType`, `ThreeTorsionPoints` and `ThreeTorsionMatrices` (which gives the translation action of each 3-torsion point on $E$ as a linear transformation on $\mathbf{P}^2$).

### 17.2.4 Integral Points

Bug Fixes:

- The reliability of the `IntegralPoints` function has been improved (previously it had failed to return all the integral points in some examples). By default, both the new and old routines are run in parallel, as a check. When the option `Fast` is set to true, only the new version of the routine is used (in many cases it is much faster).

### 17.2.5 Elliptic Curves over Number Fields

New Features:

- Local `RootNumbers` can now be computed at all primes (including primes above 2).

## 17.3 Elliptic Curves over Finite Fields

New Features:

– An implementation of Gaudry,Heß and Smart's explicit Weil descent for an elliptic curve $E$ in characteristic 2 has been included. This allows the reduction of problems in large subgroups of $E(K)$, like the Discrete Logarithm Problem, to corresponding ones in the Jacobian of $C$ over $k$ where $C$ is the (higher genus) descent curve and $k$ is a proper subfield of the original base field $K$.

– The main function for the above is `ECWeilDescent`, taking $E$ and $k$ as arguments as well as an additional parameter $c$ which determines the curve $C$. In addition to $C$, the divisor map from points of $E(K)$ (considered as degree 1 divisors) to the corresponding divisor on $C$ (as a `DivCrvElt`) is returned in the general case. If $C$ is hyperelliptic, then the divisor map returned is the actual divisor class map from $E$ to the Jacobian of $C$.

– There are additional helper functions `ECWeilDescentDegree` and `ECWeilDescentGenus` to quickly compute the degree and genus of the (plane) curve $C$ for a given $E$,$k$,$c$ input without actually performing the descent.

## 17.4 Elliptic Curves over Function Fields

This is a new package for elliptic curves with coefficients in a function field $k(C)$ where $C$ is a regular projective curve over some field $k$ (usually a number field or a finite field). The commands are largely parallel to those for elliptic curves over the rationals; one can compute local information (Tate's algorithm and so forth), a minimal model, the $L$-function, the 2-Selmer group, and the Mordell–Weil group. This goes in order of decreasing generality: local information is available for curves over univariate function fields over any exact base field, while at the other extreme Mordell–Weil groups are available only for curves over rational function fields over finite fields for which the associated elliptic surface is a rational surface. The generality of many of the commands will be expanded in future releases.

Features:

– For an elliptic curve defined over the function field of an arbitrary curve, the conductor and places of bad reduction are computed. For places of bad reduction, `LocalInformation` carries out Tate's algorithm, determining the Kodaira type and a minimal model.

– In the same generality, the Neron-Tate height of a point, and the height pairing for a given sequence of points, can be computed.

– For curves over function fields whose base field is finite (in other words, elliptic surfaces over finite fields) there is considerable functionality for counting points on the surface over finite field extensions. In this way, the `LFunction` of the elliptic curve is obtained. This in turn provides `AnalyticInformation` (conditional on the Birch–Swinnerton-Dyer or Artin-Tate conjectures), predicting the Mordell-Weil rank of the elliptic curve, the geometric rank, and the product of the regulator and the order of the Tate-Shafarevich group.

– The `TwoSelmerGroup` can be computed for an elliptic curve defined over a function field of odd characteristic whose base field is finite.

– The Mordell-Weil group can be computed for an elliptic curve defined over a *rational* function field whose base field is finite, and such that the elliptic curve, viewed as a surface over that finite field, is a rational surface. This is done by computing the Neron-Severi group of the surface, and the geometric Mordell-Weil group can also be obtained.

– The action of Frobenius on points or on fibres can be computed.

## 17.5 Genus One Models

This new package contributed by Tom Fisher deals with curves of genus 1 given by models of a special kind (genus one normal curves) of degree $2, 3, 4$ and 5. The principal functionality involves invariant theory, and applications of this to arithmetic questions.

A genus one model in Magma has type `ModelG1`, and this is not a subtype of `Crv` or `Sch`; however the defining data for models of degree $2, 3$ and 4 amounts to, respectively, a hyperelliptic curve, a plane cubic curve, and an intersection of two quadrics in $\mathbf{P}^3$. A model of degree 5 is given as a five-by-five matrix of linear forms in five variables.

Features:

- *Invariant theory:* The discriminant, and the $a$-, $b$-, and $c$-invariants of a model are computed. The `Hessian`, the `CoveringCovariants`, the `HesseCovariants`, and the `Contravariants` of a given model are also computed.
- As applications of the invariants, the `Jacobian` of a model can be computed, as well as the `nCovering` map to the Jacobian.
- Certain calculations in the Weil–Chatelet group of an elliptic curve can be performed: the sum of two models of degree 3, or multiplication by 2 for a model of degree 4 or 5.
- There is functionality to compute minimised or reduced models of a given model (except for models of degree 5).
- Families of elliptic curves that have the same Galois action on their $n$-torsion as a given elliptic curve (for $n = 2, 3, 4$ or 5) are computed by `RubinSilverbergPolynomials`. These are the same families defined by Rubin and Silverberg in *Families of elliptic curves with constant mod p representation* (in *Elliptic curves, modular forms and Fermat's last theorem*, Internat. Press, Cambridge MA, 1995).
- Conversion functions relating genus one models to the curves arising in the machinery for 2-descent and 4-descent are provided.

## 17.6 Hyperelliptic Curves

### 17.6.1 Jacobians over Number Fields

Changes:

– `TwoSelmerGroup`: Previously there were two separate routines for computing the 2-Selmer group of the Jacobian of a hyperelliptic curve, namely `TwoSelmerGroup` and `TwoSelmerGroupData`. Now, there is a single intrinsic `TwoSelmerGroup`. In cases where either of the previous intrinsics were applicable, the user may choose which is used by setting the optional parameter `Al` to `TwoSelmerGroupOld` or to `TwoSelmerGroupData`.

The new intrinsic returns an abstract group together with a map to the relevant algebra. Other data previously available can still be obtained by setting optional parameters appropriately. The upper bound on the rank which was previously the first value returned by `TwoSelmerGroupData` is now obtained by calling `RankBound` (see below).

The recommended way to control the bounds used in class group computations is to use the new feature for "globally" setting the bounds (as functions on orders in number fields), by calling `SetClassGroupBoundMaps` or the simpler alternative `SetClassGroupBounds`. These bounds will then be used by default in all class group computations during the same Magma session. They should be set before TwoSelmerGroup is called. They may be reset at any time.

The required unit group computations are now done by a call to `pSelmerGroup` in all cases (this was not previously called in `TwoSelmerGroupData`).

New Features:

– Intrinsics `RankBound` and `RankBounds` have been provided, to collect the information available from various other functions about the rank of $J(\mathbf{Q})$ for the Jacobian $J$ of a hyperelliptic curve of genus 2. `RankBound` gives an upper bound, taking into account the 2-Selmer group of $J$, the 2-torsion, and whether the order of the Tate-Shafarevich group is a square or twice a square (this assumes that the order is finite). In addition, `RankBounds` returns a lower bound obtained by a certain amount of searching for points.

Bug Fixes:

– Several improvements and fixes have been made to the local point search phase in the routines for computing 2-Selmer groups. However, it is still best to provide minimal models for the curve (or nearly minimal models, particularly at 2), especially for curves over number fields. Further improvements are expected in future patch releases.

### 17.6.2 Jacobians over Finite Fields

New Features:

– Routines for point-counting and computing zeta-functions using deformation methods have been added for parametrized families of hyperelliptic curves and their Jacobians in small, odd characteristic. These are faster than the existing Kedlaya implementation for a single curve (once the ground field becomes moderately large) and also have the advantage of being able to compute results for multiple curves in the family in less time than for the individual curves. The functions are `JacobianOrdersByDeformation`, `EulerFactorsByDeformation`, `ZetaFunctionsByDeformation`.

## 17.7 Modular Abelian Varieties

New Features:

– A package for computing "building blocks" has been contributed by Jordi Quer. A modular abelian variety $A$ can be decomposed (up to isogeny) as a power $B^r$ for some abelian variety $B$ over $\overline{Q}$, which is called a building block. The package provides tools, in the non-CM case, for determining $r$, the endomorphism algebra of $B$ (which is either a number field or a quaternion algebra over some number field) and the fields of definition of $B$. In general the fields of definition are described by an element of the Brauer group of some number field.

The intrinsics take spaces of modular symbols (with sign +1), rather than modular abelian varieties. The main intrinsics are `HasCM`, `InnerTwists`, `DegreeMap`, `BrauerClass`, which computes the class of the endomorphism algebra of $B$, and `ObstructionDescentBuildingBlock` which computes the Brauer element that describes the fields of definition of $B$.

A brief summary of the theory is provided in the handbook.

Bug Fixes:

– In `InnerTwistCharacters`, the bound 15+N div 4 has been replaced by 15+N div 2, because using the former too many twists were detected (for example, for level 28 and quadratic character, and for level 52 and characters of orders 4 and 12).

– `InnerTwistCharacters` has been changed so that the output does not contain any CM twists in the case of squarefree level and trivial Nebentypus.

# 18 Incidence Structures

## 18.1 Graphs

Bug Fixes:

– It is now possible to use the label 0 on edges and vertices of graphs.

## 18.2 Hadamard Matrices

Bug Fixes:

– Computation of canonical forms now works for matrices over rings other than **Z**.

## 18.3 Finite Incidence Geometry

New Features:

– Code written by Dimitri Leemans for testing whether a coset geometry has the Intersection Property has been included.

# 19 Coding Theory

## 19.1 Linear Codes over Finite Fields

New Features:

- The best published decoding attacks on the McEliece cryptosystem together with improved attacks have been implemented. These include attacks developed by McEliece, Lee & Brickell, Leon, Stern and Canteaut & Chabaud as well as generalized combinations of attacks.

## 19.2 Algebraic-geometric Codes

New Features:

- An efficient algorithm has been implemented for decoding algebraic-geometric codes up to the Goppa designated distance.

## 19.3 Low Density Parity Check Codes

Basic machinery has been installed for constructing and analysing Low Density Parity Check codes (LDPC codes).

Features:

- Construction of LDPC codes from sparse matrices
- Deterministic LDPC constructions
- Random constructions from regular and irregular LDPC ensembles
- Iterative LDPC decoding
- Simulation of decoding performance on specified channels
- Density evolution on binary symmetric and Gaussian channels for given channel parameters, as well as threshold determination.
- Small database of good irregular LDPC ensembles.